University of INSA Lyon

IST-4-DBM1
Telecommunications Department
Databases Part 1

Group Project:
Video Game Market

Prepared By:
Chad Long (04027054)
Jordan Ukawoko (04027217)
Luca Bova (04027050)

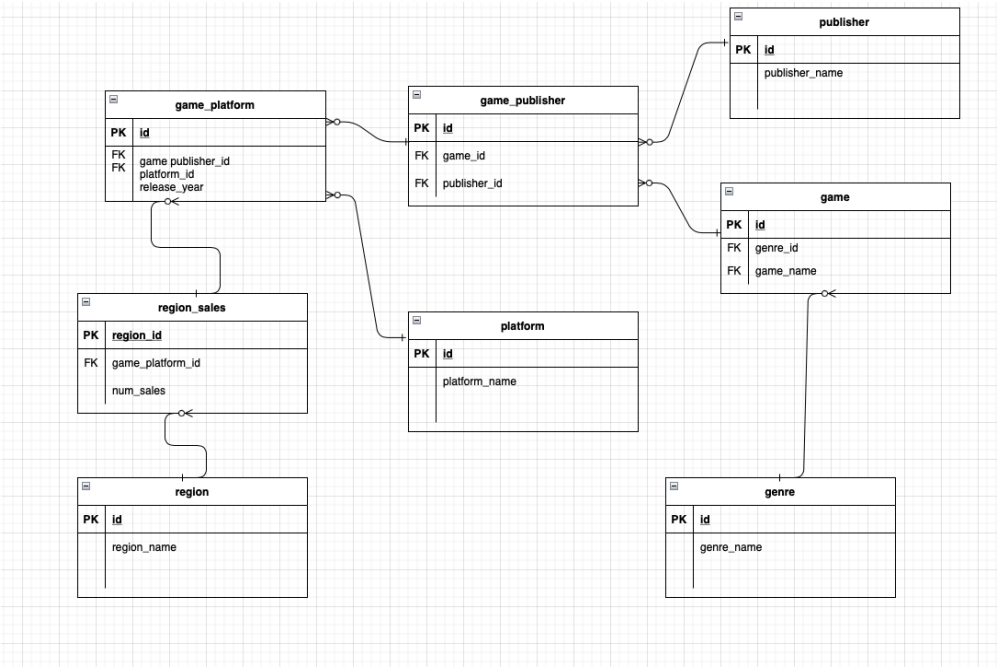
Prepared For:
Associate Professor Riccardo Tommasini

Dataset
https://www.kaggle.com/datasets/gregorut/videogamesales

Introduction - About the Dataset

This dataset is a compilation of video game sales exceeding 100,000 copies and scraped by VGChartz and pulled from Kaggle. Within this dataset, there are eleven fields ranging from the platform, release year, publisher, sales by region, and more. This dataset gives us a legitimate and robust sample size of sales from around the globe. We were able to identify eight entities, insert their primary keys, list the relationship types, and add their attributes. From constructing the entity diagram, we moved through the workflow into identifying relationships, the relational schema, and natural language queries translated into their respective relational algebra and SQL commands.

Entity Diagram



Relationships

Relations that include M:N relationships are game_platform and game_publisher. This is because a relationship exists between the game and platform and the game and publisher. An M:N relationship is a relationship that contains at least two foreign keys and as we can see above the

game_platform, game_publisher, and game relations all have at least two foreign keys thus we can conclude there is an M:N relationship between game_platform and game_publisher.

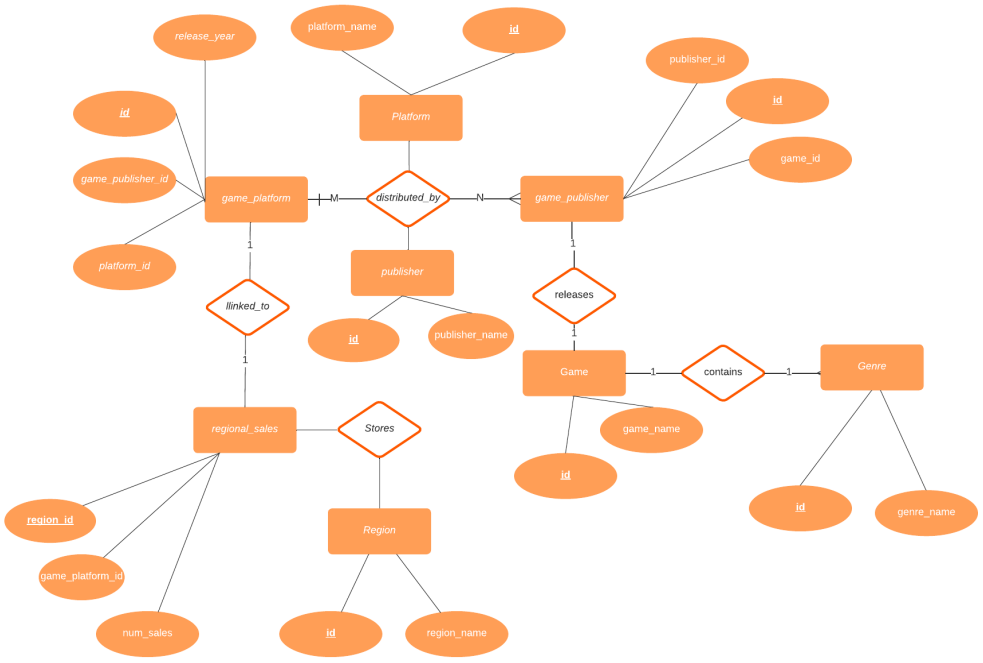
- A publisher can publish many games and they can publish games on multiple different platforms. An example includes EA publishing FIFA 23 on PS4 & Xbox One while at the same time publishing Mass Effect (a different game but the same publisher) on the Nintendo Wii & PC (same publisher but different platforms).

A one-to-one relationship is a type of cardinality that refers to the relationship between two entities A and B in which an element of A may only be linked to a component of B, and vice versa. Here we found the 1:1 relationships in our ER Model →

- The game_platform entity is related to the region_sales entity by the game_platform_id attribute
- The game_publisher entity is related to the game_platform entity by the publisher_id
- The genre entity is related to the game entity by the genre_id attribute
- The game_publisher entity is related to the game entity by the game_id attribute

ER DIAGRAM

This is our ER Diagram. Just like the Entities Model diagram.



- The game_platform entity is related to the region_sales entity by the game_platform_id attribute
- The game_publisher entity is related to the game_platform entity by the publisher_id
- The genre entity is related to the game entity by the genre_id attribute
- The game_publisher entity is related to the game entity by the game_id attribute
- There is a ternary relationship between game_platform, Platform, and game_publisher
- Binary relationship between regional_sales and game_platform
- Binary relationship between game_publisher and Game
- Binary Relationship between Game and Genre
- Binary Relationship between regional_saes and Region

A M:N relationship also exists between game_platform and game_publisher. An ER Diagram is used to design or debug relational databases in the fields such as Software Engineering and Business Information Systems.

- 8 of the entities are strong as they contain primary keys

ID is the primary key for 8 of these entities. A primary key can be defined as "which attributes identify a record," and in simple cases constitute a single attribute: a unique ID. As you can see the ID attribute is unique to all of these entities.

Foreign keys also exist within this relationship schema. Examples of foreign keys are

- game_publisher_id and game_platform_id in the game_platform entity
- game_id and publisher_id in the game_publisher entity
- genre_id and game_name in the game entity
- game_platform_id in the region_sales entity

These are foreign keys because a foreign key is a field (or collection of fields) in one table, that refers to the primary key in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

For example - **game_platform_id** is a foreign key in the region_sales entity because **id** is the primary key in the game_platform entity.

Natural Language Queries

Below are three examples of queries expressed in natural language, a full list of all the queries and relational algebra are in their own files in the repository:

1. How many games were released in 2016?
2. Group the games from the highest selling to the lowest
3. Find the average number of sales of games released in North America

We show examples of these queries in SQL & Relational Algebra, down below.

```
SELECT COUNT (release_year) FROM game_pl
WHERE release_year = 2016
```

```
SELECT
g.game_name,
pl.platform_name,
gp.release_year,
pub.publisher_name,
SUM(rs.num_sales) AS global_sales
FROM regional_sales rs
INNER JOIN region r ON rs.region_id = r.id
INNER JOIN game_pl gp ON rs.game_platform_id = gp.id
INNER JOIN game_publisher gpub ON gp.game_publisher_id = gpub.id
INNER JOIN games g ON gpub.game_id = g.id
INNER JOIN platform pl ON gp.platform_id = pl.id
INNER JOIN publisher pub ON gpub.publisher_id = pub.id
GROUP BY g.game_name, pl.platform_name, gp.release_year, pub.publisher_name
ORDER BY SUM(rs.num_sales) DESC;
```

```
SELECT
AVG(rs.num_sales) AS north_america_avg
FROM regional_sales rs
WHERE region_id = 1
```

Relational Algebra

$$\pi_{\text{COUNT (release\_year)}} \gamma_{\text{COUNT (release\_year)}} \sigma_{\text{release\_year = 2016}} \text{game\_pl}$$

$$\tau_{\text{SUM (num\_sales) ↓}} \pi_{\text{g . game\_name, pl . platform\_name, gp . release\_year, pub . publisher\_name, SUM (num\_sales) → global\_sales}} \gamma_{\text{game\_name, platform\_name, release\_year, publisher\_name, SUM (num\_sales)}} (\rho_{rs}\text{regional\_sales} \bowtie_{\text{rs . region\_id = r . id}} \rho_{r}\text{region} \bowtie_{\text{rs . game\_platform\_id = gp . id}} \rho_{gp}\text{game\_pl} \bowtie_{\text{gp . game\_publisher\_id = gpub . id}} \rho_{gpub}\text{game\_publisher} \bowtie_{\text{gpub . game\_id = g . id}} \rho_{g}\text{games} \bowtie_{\text{gp . platform\_id = pl . id}} \rho_{pl}\text{platform} \bowtie_{\text{gpub . publisher\_id = pub . id}} \rho_{pub}\text{publisher})$$

$$\pi_{\text{AVG (num\_sales) → north\_america\_avg}} \gamma_{\text{AVG (num\_sales)}} \sigma_{\text{region\_id = 1}} \rho_{rs}\text{regional\_sales}$$

Normalization & Relationship Schema

Normalization is the process of eliminating data redundancy and enhancing data integrity in the table.
Normalization also helps to organize the data in the database. The goal is to translate our schemas into first form, second form, and third form. This was done for ease of reading and access, as well as to reduce the overall content and make it easier to work with. Normalizing the content also made working with it and making diagrams with it easier.

First Form

game_platform, regional_sales, region, game_publisher, platform, genre, game, publisher, stores, distributed_by, linked_to, releases, contains

- The first normal form makes each attribute in the table a single-valued attribute

Second Form

game_platform(id), regional_sales(region_id), region(id), game_publisher(id), platform(id), genre(id), game(id), publisher(id), stores, distributed_by, linked_to, releases, contains

As you can see in the second form we are listing all the entities but only with their **primary keys.** The second form makes the table with no partial dependencies.

Third Form

game_platform(**id**, game_publisher_id, platform_id, release_year)

game_publisher(**id**, game_id, publisher_id)

publisher(**id**, publisher_name)

region_sales(**region_id**, game_platform_id, num_sales)

platform(**id**, platform_name)

region(**id**, region_name)

genre(**id**, genre_name)

game(**id**, genre_id, game_name)

stores(region,regional_sales)

distributed_by(publisher, platform,game_publisher,game_platform)

linked_to(regional_sales, game_platform)

releases(game, game_publisher)

contains(game, genre)

- As you can see in the third form we are listing all the entities with all their attributes
- The third form makes the table with no transitive dependency

HTML Website

We decided to create a very small HTML website to display our queries. We thought it would be interesting to create a small HTML application, to visualize some of the queries run in PostgreSQL. To access the HTML website, the user must click on the index.html from the "HTML Application" Folder. We also have a "quries.sql" file which includes all the queries we came up with. Here is an image showing a snippet of our HTML Website.

Concluding Thoughts

This database was interesting to work with. Brainstorming the entities for the ER diagram and how they would flow together was a fun process. We wanted to utilize a dataset that was interesting to us as well as easy to understand but with lots of data, and we found that with the global sales of video games. The most difficult step was implementing and explaining the Natural Language Queries. This was a relatively difficult step to fully implement. Understanding

how to extrapolate what we needed from the databases through natural language instead of relational algebra or standard commands was very challenging. However, the results will speak for themselves, as the full grammar is shown within the presentation. This was a very difficult yet rewarding project. Learning to utilize different schemas for different purposes and the two different types of diagrams was engaging.